Assignment 2 – Ruby

This assignment consists of 3 problems. You must solve each problem then finally upload the solution codes to the "Ruby" dropbox on avenue.

Problem 1

Objectives

- Properly tests for object equality
- Properly tests for a `nil` value
- Properly tests for a boolean `true/false` value
- Implements these tests using a case statement

Functional Requirements

1. Re-write the following if/else statement using a case statement in Ruby

```
some_var = "false"
another_var = "nil"

if some_var == "pink elephant"
  puts "Don't think about the pink elephant!"

elsif another_var.nil?
  puts "Question mark in the method name?"

elsif some_var == false
  puts "Looks like this one should execute"

else
  puts "I guess nothing matched... But why?"
end
```

2. Analyze the outcome of the original script and state the reasons why each of the first three tests fail and how they could be made to succeed
3. Change the source code of the solution to verify your conclusions

Getting Started

1. Download and extract the starter set of files. The root directory is **module3_exercise1**.
    - **module3_exercise1.rb** – contains the starting if/else statements. Your solution must be placed within this file.

- **spec** – this directory contains tests to verify your solution. You should not modify anything in this directory
- **.rspec** – configuration gems used by the rspec unit tests. If you move your files you must take care to also copy this file

2. Install the following gems used by the rspec unit tests. You may have some of these already installed

```
gem install rspec
gem install rspec-its
```

3. Run the provided Ruby script containing the if/else
4. Run the rspec command to execute the unit tests within the spec directory. This command should be run from the root directory of the project. This should result in several failures until you complete your solution
5. Implement the solution and retest

Technical Requirements

1. Implement all parts of this exercise within the **module3_ecercise1.rb** file in the root directory.
2. Remove all traces of if, elsif
3. Re-implement the solution in terms of case and associated constructs
4. Return the same result as the initial solution after applying the case statement
5. Archive the directory in a zip archive; module3_exercise1 to be uploaded to Avenue dropbox

Problem 2

Objectives

- manipulate collections to derive a result

Functional Requirements

1. Given the following collection examples ….

```
# Grab 23 random elements between 0 and 10000
arr = (1..10000).to_a.sample(23)
p arr

# This selects only elements that when divided by 3 have a remainder of 0
# using the % (modulus) operator
p arr.select { |element| element % 3 == 0 }

# Using `reject` method filter out anything less than 5000
# and use `sort` and `reverse` methods to sort in descending order
# Start with the line below and continue as 1 long method chain
# p arr.select { |element| element % 3 == 0 }
```

2. Write a single chain of commands to find all numbers that
    - are from an array of numbers 1..10000 inclusive
    - are divisible by 3
    - are not less than 5000
    - sorted in reverse order

Getting Started

6. Download and extract the starter set of files. The root directory is **module3_exercise2**.
    - **module3_exercise2.rb** – contains the starting examples. Your solution must be placed within this file.
    - **spec** – this directory contains tests to verify your solution. You should not modify anything in this directory
    - **.rspec** – configuration gems used by the rspec unit tests. If you move your files you must take care to also copy this file
7. Install the following gems used by the rspec unit tests. You may have some of these already installed

```
gem install rspec
gem install rspec-its
```

8. Run the provided Ruby script containing the if/else
9. Run the rspec command to execute the unit tests within the spec directory. This command should be run from the root directory of the project. This should result in several failures until you complete your solution
10. Implement the solution and retest

Technical Requirements

1. Implement all parts of this exercise within the **module3_ecercise2.rb** file in the root directory.
2. Your script must use a single chain of commands that locate numbers based on the functional requirements above
3. Your answer must be printed as an array of numbers using the p print command as the last line of the script output
4. Archive the directory in a zip archive; module3_exercise1 to be uploaded to Avenue dropbox

Problem 3

Objectives

- implement a Ruby class with – class attribute(s) – class method(s) – initializer method(s) – instance attribute(s) – instance method(s)

Functional Requirements

1. Write a `Person` class with a

- `first_name`
- `last_name`

2. Track each instance of `Person`
3. Search for a `Person` by `last_name`

Getting Started

1. Download and extract the starter set of files. The root directory is **module3_exercise3**.
    - **module3_exercise3.rb** – contains the starting examples. Your solution must be placed within this file.
    - **spec** – this directory contains tests to verify your solution. You should not modify anything in this directory
    - **.rspec** – configuration gems used by the rspec unit tests. If you move your files you must take care to also copy this file
2. Install the following gems used by the rspec unit tests. You may have some of these already installed

```
gem install rspec
gem install rspec-its
```

3. Run the provided Ruby script containing the if/else
4. Run the rspec command to execute the unit tests within the spec directory. This command should be run from the root directory of the project. This should result in several failures until you complete your solution
5. Implement the solution and retest

Technical Requirements

1. Implement all parts of this exercise within the **module3_ecercise3.rb** file in the root directory.
2. Your script must contain a `Person` class
3. The `Person` class must have
    - a `first_name` and `last_name` attribute with public accessors for setting and getting the attributes
    - a class attribute called `people` that holds an array of objects
    - an `initialize` method to initialize each instance
    - a `to_s` method to return a formatted string of the person's name
    - a search method to locate all people with a matching last_name
4. The `Person initialize` method must
    - accept two parameters (`first_name` and `last_name`) and use them to initialize the `first_name` and `last_name` instance attributes
    - insert the created instance (self) into the people class variable
5. The `Person to_s` instance method must
    - return a formatted string as `first_name(space)last_name`
6. The `Person search` class method must

- accept a `last_name` parameter
- search the `people` class attribute for instances with the same `last_name`
- return a collection of matching instances

7. Archive the directory in a zip archive; module3_exercise1 to be uploaded to Avenue dropbox

Rubric

Each of these problems is either marked correct or wrong; there are no partial marks.

Criterion	Points (out of 15)
Problem 1	5
Problem 2	5
Problem 3	5